Back-End, Web / Software Developer

Round-2

Questions Bank

Questions about Design Patterns:

- Why are global and static objects evil? Can you show it with a code example?
- Tell me about Inversion of Control and how does it improve the design of code.
- The Law of Demeter (the Principle of Least Knowledge) states that each unit should have only limited knowledge about other units and it should only talk to its immediate friends (sometimes stated as "Don't talk to strangers"). Would you write code violating this principle, show why it is a bad design and then fix it?
- Active-Record is the design pattern that promotes objects to include functions such as Insert, Update, and Delete, and properties that correspond to the columns in some underlying database table. In your opinion and experience, which are the limits and pitfalls of the this pattern?
- Data-Mapper is a design pattern that promotes the use of a layer of Mappers that moves data between objects and a database while keeping them independent of each other and the mapper itself. On the contrary, in Active-Record objects directly incorporate operations for persisting themselves to a database, and properties corresponding to the underlying

database tables. Do you have an opinion on those patterns? When would you use one against the other?
- Why it is often said that the introduction of `null` is a "Billion dollar mistake"? Would you discuss the techniques to avoid it, such as the Null Object Pattern introduced by the GOF book, or Option types?
- Many state that, in Object-Oriented Programming, Composition is often a better option than Inheritance. What's you opinion?
- What is an Anti-corruption Layer?
- Singleton is a design pattern that restricts the instantiation of a class to one single object. Writing a Thread-Safe Singleton class is not so obvious. Would you try?
- The ability to change implementation without affecting clients is called Data Abstraction. Produce and example violating this property, then fix it.
- Write a snippet of code violating the Don't Repeat Yourself (DRY) principle. Then, fix it.
- How would you deal with Dependency Hell?
- Is goto evil? You may have heard of the famous paper "Go To Statement Considered Harmful" by Edsger Dijkstra, in which he criticized the use of the `goto` statement and advocated structured programming instead. The use of `goto` has always been controversial, so much that even Dijkstra's letter was criticized with articles such as "'GOTO Considered Harmful' Considered Harmful". What's your opinion on the use of `goto`?
- The robustness principle is a general design guideline for software that recommends "*Be conservative in what you send, be liberal in what you accept*". It is often reworded as "*Be a tolerant reader and a careful writer*". Would you like to discuss the rationale of this principle?
- Separation of Concerns is a design principle for separating a computer program into distinct areas, each of ones

addressing a separate concern. There are a lot of different mechanisms for achieving Separation of Concerns (use of objects, functions, modules, or patterns such as MVC and the like). Would you discuss this topic?

Questions about Code Design:

- It is often heard that one of the most important goals in Object-Oriented Design (and code design in general) is to have High Cohesion and Loose Coupling. What does it mean? Why is it that important and how is it achieved?
- Why does array index start with '0' in most of languages?
- How do tests and TDD influence code design?
- Write a snippet of code violating the Don't Repeat Yourself (DRY) principle. Then, explain why is it a bad design, and fix it.
- What's the difference between cohesion and coupling?
- What is refactoring useful for?
- Are comments in code useful? Some say they should be avoided as much as possible, and hopefully made unnecessary. Do you agree?
- What is the difference between design and architecture?
- Why in TDD are tests written before code?
- C++ supports multiple inheritance, and Java allows a class to implement multiple interfaces. What impact does using these facilities have on orthogonality? Is there a difference in impact between using multiple inheritance and multiple interfaces? Is there a difference between using delegation and using inheritance? [This question is from The Pragmatic Programmer, by Andrew Hunt and David Thomas]
- Pros and cons of holding domain logic in Stored Procedures.

- In your opinion, why have Object-Oriented Design dominated the market for so many years?
- What would you do to understand if your code has a bad design?

Questions about Languages:

- Tell me the 3 worse defects of your preferred language
- Why is there a rising interest on Functional Programming?
- What is a closure, and what is useful for? What's in common between closures and classes?
- What are generics useful for?
- What are high-order functions? What are they useful for? Write one, in your preferred language.
- Write a loop, then transform it into a recursive function, using only immutable structures (i.e. avoid using variables). Discuss.
- What does it mean when a language treats functions as first-class citizens?
- Show me an example where an Anonymous Function can be useful
- There are a lot of different type systems: let's talk about static and dynamic type systems, and about strong and weak ones. You surely have an opinion and a preference about this topic. Would you like to share them, and discuss why and when would you promote one particular type system for developing an enterprise software?
- What are namespaces useful for? Invent an alternative.
- Talk about Interoperability between Java and C# (in alternative, choose 2 other arbitrary languages)

- Why do many software engineers not like Java?
- What makes a good language good and a bad language bad?
- Write two functions, one Referentially Transparent and the other one Referentially Opaque. Discuss.
- Whats the Stack and what's the Heap? What's a Stack Overflow?
- Why is it important that in a language functions are first class citizens?
- Some languages, especially the ones that promote a Functional approach, allow a technique called Pattern Matching. Do you know it? How is Pattern Matching different from Switch clauses?
- Why do some languages have no exceptions by design? What are the pros and cons?
- If `Cat` is an `Animal`, is `TakeCare<Cat>` a `TakeCare<Animal>`?
- Why in Java, C# and many other languages constructors are not part of the interface?
- In the last years there has been a lot of hype on Node. What's your opinion on the use in the back end of a language that was initially conceived to run in the browser?
- Pretend you have a time machine and pretend that you have the opportunity to go to a particular point in time during Java's (or C#, Python, Go or whatever) history, and talk with some of the JDK architects. What would you try to convince them of? Removing checked exceptions? Adding unsigned primitives? Adding multiple-inheritance?

Questions about Web development:

- Why first-party cookies and third-party cookies are treated so differently?
- How would you manage Web Services API versioning?
- From a Back End perspective, are there any disadvantages or drawbacks on the adoption of Single Page Applications?
- Why do we usually put so much effort for having stateless services? What's so good in stateless code and why and when statefulness is bad?
- REST and SOAP: when would you choose one, and when the other?
- In Web development, Model-View Controller and Model-View-View-Model approaches are very common, both in the Back End and in the Front End. What are they, and why are they advisable?

Questions about Databases:

- How would you migrate an application from a database to another, for example from MySQL to PostgreSQL? If you had to manage that project, which issues would you expect to face?
- Why databases treat null as a so special case? For example, why in SQL `SELECT * FROM table WHERE field = null` does not match records with null `field`?
- ACID is an acronym that refers to Atomicity, Consistency, Isolation and Durability, 4 properties guaranteed by a database transaction in most of the database engines. What do you know about this topic? Would you like to elaborate?
- How would you manage database schema migrations, that is, how would you automate the changes a database

schema is affected to, as the application evolve, version after version?

- How is Lazy Loading achieved? When is it useful? What are its pitfalls?
- The so called "N + 1 problem" is an issue that occurs when the code needs to load the children of a parent-child relationship with a ORMs that have lazy-loading enabled, and that therefore issue a query for the parent record, and then one query for each child record. How to fix it?
- How would you find the most expensive queries in an application?
- In your opinion, is it always needed to use database normalization? When is it advisable to use denormalized databases?
- Of of the Continuous Integration's techniques is called Blue-Green Deployment: it consists in having two production environments, as identical as possible, and in performing the deployment in one of them while the other one is still operating, and than in safely switching the traffic to the second one after some convenient testing. This technique becomes more complicated when the deployment includes changes to the database structure or content. I'd like to discuss this topic with you.

Questions about NoSQL:

- What is Eventual Consistency?
- The Brewer's Theorem, most commonly known as the CAP theorem, states that in the presence of a Network Partition (the P in CAP), a system's designer has to choose between Consistency (the C in CAP) and Availability (the A in CAP). Can you think about examples of CP, AP and CA systems?
- How would you explain the recent rise in interest for NoSQL?
- How does NoSQL tackle scalability challenges?
- In which case would you use a document database like MongoDB instead of a relational database like MySQL or PostgreSQL?

Questions about code versioning:

- Why is branching with Mercurial or git easier than with SVN?
- What are the pros and cons of Distributed Version Control Systems like Git over Centralized ones like SVN?
- Could you describe GitHub Flow and GitFlow workflows?
- What's a rebase?
- Why merges are easier with Mercurial and git than with SVN and CVS?

Questions about Concurrency:

- Why do we need Concurrency, anyway? Explain.
- Why is testing multithreading / concurrent code so difficult?
- What is a Race Condition? Code an example, using whatever language you like.
- What is a Deadlock? Would you be able to write some code that is affected by deadlocks?

- What is Process Starvation? If you need, let's review its definition.
- What is a Wait Free algorithm?

Questions about Distributed Systems:

- How to test a distributed system?
- In which case would you apply asynchronously communication between two systems?
- What are the general pitfalls of Remote Procedure Call?
- If you are building a distributed system for scalability and robustness, what are the different things you'd think of in the case you are working in a closed and secure network environment or in geographically distributed and public system?
- How to manage Fault Tolerance in a Web application? And in a Desktop one?
- How to deal with failures in Distributed Systems?
- Let's talk about the several approaches to Reconciliation after network partitions
- What are the Fallacies of Distributed Computing?
- When would you use Request/Reply and when Publish/Subscribe?
- Suppose the system you are working on does not support transactionality. How would you implement it from scratch?

Questions about Software Lifecycle and Team Management:

- What is agility?
- How would you deal with Legacy Code?
- Say I'm your Project Manager, and I'm no expert in programming. Would you try explaining me what Legacy Code is and why should I care about code quality?
- I'm the CEO of your company. Explain to me Kanban and convince me to invest in it.
- What is the biggest difference between Agile and Waterfall?
- Being a team manager, how would you deal with the problem of having too many meetings?
- How would you manage a very late project?
- "*Individuals and interactions over processes and tools*" and "*Customer collaboration over contract negotiation*" comprise half of the values of the Agile Manifesto. Discuss
- Tell me what decisions would you take if you could be the CTO of your Company.
- Are Program Managers useful?
- Organize a development team using flexible schedules (that is, no imposed working hours) and "Take as you need" vacation policy
- How would you manage a very high turn over and convince developers not to leave the team, without increasing compensation? What could a Company improve to make them stay?
- What are the top 3 qualities you look for in colleagues, beyond their code?
- What are the top 3 things you wish non-technical people knew about code?
- Imagine your company gives you 1 month and some budget to improve your and your colleagues' daily life. What would you do?

Questions about logic and algorithms:

- Make a FIFO Queue using only LIFO Stacks. Then build a LIFO Stack using only FIFO Queues.
- Write a snippet of code affected by a Stack Overflow
- Write a tail-recursive version of the factorial function
- Using your preferred language, write a REPL that echoes your inputs. Evolve it to make it an RPN calculator.
- How would you design a "defragger" utility?
- Write a program that builds random mazes.
- Write a sample code that produces a memory leak
- Generate a sequence of unique random numbers
- Write a simple Garbage collection system
- Write a basic message broker, using whatever language you like.
- Write a very basic web server. Draw a road map for features to be implemented in the future.
- How would you sort a 10GB file? How would your approach change with a 10TB one?
- How would you programmatically detect file duplicates?

Questions about Software Architecture:

- When is a cache not useful or even dangerous?
- Why does Event-Driven Architecture improve scalability?
- What makes code readable?
- What is the difference between emergent design and evolutionary architecture?
- Scale out vs scale up: how are they different? When to apply one, when the other?
- How to deal with failover and user sessions?

- What is CQRS (Command Query Responsibility Segregation)? How is it different from the oldest Command-Query Separation Principle?
- The so called "multitier architecture" is an approach to design a client–server system aimed to keep physically and logically separated presentation, application processing, data management and other functions. The most widespread of the multitier architectures is the three-tier architecture. Would you discuss the pros and cons of such approach?
- How would you design a software system for scalability?
- Someone gave the name "The "C10k problem" to the problem of optimising network sockets to handle over 10.000 open connections at once. While handling 10.000 concurrent clients is not the same as handling 10.000 open connection, the context is similar. It's a tough challenge anyway, and no one is expected to know every single detail to solve it. It may be interesting to discuss the strategies you know to deal with that problem. Would you like to try?
- How would you design a decentralized (that is, with no central server) P2P system?
- You may recall that Common Gateway Interface (CGI) is a standard protocol for web servers to execute programs (CGI scripts) that execute as Command-line programs on a server, and that dynamically generate HTML pages when invoked by a HTTP request. Perl and PHP used to be common languages for such scripts. In CGI, a HTTP request generally causes the invocation of a new process on the server, but FastCGI, SCGI and other approaches improved the mechanism, raising the performance, with techniques such as preforking processes. Can you imagine why has't CGI eventually win, and was instead replaced with other architectural approaches?
- How would you defend the design of your systems against Vendor Lock-in?

- What are the disadvantages of the Publish-Subscribe pattern at scale?
- What's new in CPUs since the 80s, and how does it affect programming?
- In which part of the lifecycle of a software performance should be taken in consideration, and how?
- How could a Denial of Service arise not maliciously but for a design or architectural problem?
- What’s the relationship between Performance and Scalability?
- When is it OK (if ever) to use tight coupling?
- What characteristic should a system have to be Cloud Ready?
- Does unity of design imply an aristocracy of architects? Putting it simple: can good design emerge from a collective effort of all developers?
- What's the difference between design, architecture, functionality and aesthetic? Discuss.

Questions about Service Oriented Architecture and Microservices:

- Why, in a SOA, long-lived transactions are discouraged and Sagas are suggested instead?
- What are the differences between Soa and Microservices?
- Let's talk about web services versioning, version compatibility and breaking changes.
- What's the difference between a transaction and a compensation operation in a saga, in SOA?
- When is a Microservice too micro?
- What are the pros and cons of MicroService architecture?

Questions about Security:

- How do you write secure code? In your opinion, is it one of the developer's duties, or does it require a specialized role in the company? And why?
- Why is it said that cryptography is not something you should try to invent or design yourself?
- What is two factor authentication? How would you implement it in an existing web application?
- If not carefully handled, there is always a risk of logs containing sensitive information, such as passwords. How would you deal with this?
- Write down a snippet of code affected by SQL Injection and fix it.
- How would it be possible to detect SQL Injection via static code analysis? I don't expect you to write an algorithm capable of doing this, as it is probably a huge topic, but let's discuss a general approach.
- What do you know about Cross-Site Scripting? If you don't remember it, let's review online its definition and let's discuss about it.
- What do you know about Cross-Site Forgery Attack? If you don't remember it, let's review online its definition and let's discuss about it.
- How does HTTPS work?
- What's a Man-in-the-middle Attack, and why does HTTPS help protect against it?
- How can you prevent the user's session from being stolen? Chances are you remember what Session or Cookie Hijacking is, otherwise let's read its Wikipedia page together.

General Questions:

- Why does Functional Programming matter? When should a functional programming language be used?
- How do companies like Microsoft, Google, Opera and Mozilla profit from their browsers?
- Why does opening a TCP socket have a large overhead?
- What is Encapsulation important for?
- What is a real-time system and how is it different from an ordinary system?
- What's the relationship between real-time languages and heap memory allocation?
- Immutability is the practice of setting values once, at the moment of their creation, and never changing them. How can immutability help write safer code?
- What are the pros and cons of mutable and immutable values.
- What's the Object-Relational impedance mismatch?
- Which principles would you apply to define the size of a cache?
- What's the difference between TCP and HTTP?
- What are the tradeoffs of client-side rendering vs. server-side rendering?
- How could you develop a reliable communication protocol based on a non-reliable one?
- Tony Hoare who invented the null reference once said "*I call it my billion-dollar mistake*" since it lead to "*innumerable errors, vulnerabilities, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years*". Imagine you want to remove the possibility to have null references in your preferred language: how would you achieve this goal? What consequences could this have?

Open Questions:

- Why do people resist change?
- Explain threads to your grandparents
- As a software engineer you want both to innovate and to be predictable. How those 2 goals can coexist in the same strategy?
- What makes good code good?
- Explain streaming and how you would implement it.
- Say your Company gives you one week you can use to improve your and your colleagues' lifes: how would you use that week?
- What did you learn this week?
- There is an aesthetic element to all design. The question is, is this aesthetic element your friend or your enemy?
- List the last 5 books you read.
- How would you introduce Continuous Delivery in a successful, huge company for which the change from Waterfall to Continuous Delivery would be not trivial, because of the size and complexity of the business?
- When does it make sense to reinvent the wheel?
- Let's have a conversation about "*Reinventing the wheel*", the "*Not Invented Here Syndrome*" and the "*Eating Your Own Food*" practice
- What's the next thing you would automate in your current workflow?
- Why is writing software difficult? What makes maintaining software hard?
- Would you prefer working on Green Field or Brown Field projects? Why?
- What happens when you type google.com into your browser and press enter?
- What does an Operating System do when it has got no custom code to run, and therefore it looks idle? I would like

to start a discussions about interrupts, daemons, background services, polling, event handling and so on.
- Explain Unicode/Database Transactions to a 5 year old child.
- Defend the monolithic architecture.
- What does it mean to be a "Professional Developer"?
- Is developing software an art, a craftsmanship or an engineering endeavour? Your opinion.
- "People who like this also like... ". How would you implement this feature in an e-commerce shop?
- Why are corporations slower than startups in innovating?
- What have you achieved recently that you are proud of?

Questions about snippets of code:

- What's the output of this Javascript function?

```
function hookupevents() {
  for (var i = 0; i < 3; i++) {
    document.getElementById("button" + i)
      .addEventListener("click", function() {
        alert(i);
      });
  }
```

```
}
```

- About Type Erasure, what's the output of this Java snippet, and why?

```
ArrayList<Integer> li = new ArrayList<Integer>();
ArrayList<Float> lf = new ArrayList<Float>();
if (li.getClass() == lf.getClass()) // evaluates to true
  System.out.println("Equal");
```

- Can you spot the memory leak?

```
public class Stack {
    private Object[] elements;
    private int size = 0;
    private static final int DEFAULT_INITIAL_CAPACITY =
16;

    public Stack() {
        elements = new Object[DEFAULT_INITIAL_CAPACITY];
    }

    public void push(Object e) {
        ensureCapacity();
        elements[size++] = e;
    }

    public Object pop() {
        if (size == 0)
            throw new EmptyStackException();
        return elements[--size];
    }

    /**
     * Ensure space for at least one more element,
roughly
```

```
     * doubling the capacity each time the array needs to
grow.
     */
    private void ensureCapacity() {
        if (elements.length == size)
            elements = Arrays.copyOf(elements, 2 * size +
1);
    }
}
```

- `if`s and in general conditional statements lead to procedural and imperative programming. Can you get rid of this `switch` and make this snippet more object oriented?

```
public class Formatter {

    private Service service;

    public Formatter(Service service) {
        this.service = service;
    }

    public String doTheJob(String theInput) {
        String response = service.askForPermission();
        switch (response) {
        case "FAIL":
            return "error";
        case "OK":
            return String.format("%s%s", theInput,
theInput);
        default:
            return null;
        }
    }
}
```

- Can you get rid of these `if`s and make this snippet of code more object oriented?

```
public class TheService {
    private final FileHandler fileHandler;
    private final FooRepository fooRepository;

    public TheService(FileHandler fileHandler,
FooRepository fooRepository) {
        this.fileHandler = fileHandler;
        this.fooRepository = fooRepository;
    }

    public String Execute(final String file) {

        final String rewrittenUrl =
fileHandler.getXmlFileFromFileName(file);
        final String executionId =
fileHandler.getExecutionIdFromFileName(file);

        if (executionId.equals("") ||
rewrittenUrl.equals("")) {
            return "";
        }

        Foo knownFoo =
fooRepository.getFooByXmlFileName(rewrittenUrl);

        if (knownFoo == null) {
            return "";
        }

        return knownFoo.DoThat(file);
    }
}
```

- How to refactor this code?

```
function()
{
    HRESULT error = S_OK;

    if(SUCCEEDED(Operation1()))
    {
        if(SUCCEEDED(Operation2()))
        {
            if(SUCCEEDED(Operation3()))
            {
                if(SUCCEEDED(Operation4()))
                {
                }
                else
                {
                    error = OPERATION4FAILED;
                }
            }
            else
            {
                error = OPERATION3FAILED;
            }
        }
        else
        {
            error = OPERATION2FAILED;
        }
    }
    else
    {
        error = OPERATION1FAILED;
    }
```

```
    return error;
}
```

Mindcrack Questions:

This section collects some weird questions along the lines of the Manhole Cover Question.

- What would happen if you put a mirror in a scanner?
- Imagine there's a perfect clone of yourself. Imagine that that clone is your boss. Would you like to work for him/her?
- Interview me
- Why are Quora's answers better than Yahoo Answers' ones?

- Let's play a game: defend Cobol against modern languages, and try to find as many reasonable arguments as you can.
- Where will you be in 10 years?
- You are my boss and I'm fired. Inform me.
- I want to refactor a legacy system. You want to rewrite it from scratch. Argument. Then, switch our roles.
- Your boss asks you to lie to the Company. What's your reaction?
- If you could travel back in time, which advice would you give to your younger self?